Комп‘ютерний практикум № 4. LINQ to Objects

Мета:

- ознайомитися з обробкою даних з використанням бібліотеки LINQ to Objects

Теоретичні основи

Запит LINQ (Language-Integrated Query) являє собою вираз, за допомогою якого можна отримувати дані з будь-яких джерел даних.

LINQ – це модель для роботи з даними з різних видів джерел даних і в різних форматах. Всі операції запиту LINQ можна розбити на три різні групи:

1) Отримання джерела даних.
2) Створення запиту.
3) Виконання запиту.

Джерело даних для виконання запитів LINQ повинно підтримувати інтерфейс IEnumerable(Of T), де T – тип елементів джерела даних.

В якості джерела даних можуть виступати масиви (списки), бази даних та XML-документ.

Запит вказує, яку інформацію потрібно вибрати з джерела або джерел даних. При необхідності запит також вказує спосіб сортування і групування. Запит зберігається в змінній запиту та ініціалізується виразом запиту.

В C# використовується синтаксис:

тип змінної запиту = **from** змінна діапазону **in** джерело
where умова відбору даних
group групування
orderby сортування
select значення, яке повертається;

Частини where, group, orderby можуть бути відсутніми.

В якості типу може використовуватись тип IEnumerable<…> або ключове слово var.

Частина запиту from вказує джерело даних, where застосовує фільтр, а select вказує тип елементів, які повертаються. У LINQ змінна запиту зберігає відомості, необхідні для надання результатів при подальшому виконанні запиту.

Фактичне виконання запиту відкладається до виконання ітерації змінної запиту в операторі foreach. Цю концепцію називають відкладеним виконанням.

Змінні запитів LINQ визначені як IEnumerable (Of T) або як похідний тип, наприклад IQueryable (Of T).

Приклад коду

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace SimpleLINQ
{
    class Program
    {
        /// <summary>
        /// Класс данных
        /// </summary>

        public class Data
        {
            /// <summary>
            /// Ключ
            /// </summary>
            public int id;

            /// <summary>
            /// Для группировки
            /// </summary>
            public string grp;

            /// <summary>
            /// Значение
            /// </summary>
            public string value;

            /// <summary>
            /// Конструктор
            /// </summary>
            public Data(int i, string g, string v)
            {
                this.id = i;
                this.grp = g;
                this.value = v;
            }

            /// <summary>
            /// Приведение к строке
            /// </summary>
            public override string ToString()
            {
                return "(id=" + this.id.ToString() + "; grp=" + this.grp + "; value=" +
this.value + ")";
            }
        }

        /// <summary>
        /// Класс для сравнения данных
        /// </summary>
        public class DataEqualityComparer : IEqualityComparer<Data>
        {
            public bool Equals(Data x, Data y)
            {
                bool Result = false;
                if (x.id == y.id && x.grp == y.grp && x.value == y.value)
                    Result = true;
```

```
            return Result;
        }
        public int GetHashCode(Data obj)
        {
            return obj.id;
        }
    }

    /// <summary>
    /// Связь между списками
    /// </summary>
    public class DataLink
    {
        public int d1;
        public int d2;
        public DataLink(int i1, int i2)
        {
            this.d1 = i1;
            this.d2 = i2;
        }
    }

    //Пример данных
    static List<Data> d1 = new List<Data>()
    {
        new Data(1, "group1", "11"),
        new Data(2, "group1", "12"),
        new Data(3, "group2", "13"),
        new Data(5, "group2", "15")
    };

    static List<Data> d2 = new List<Data>()
    {
        new Data(1, "group2", "21"),
        new Data(2, "group3", "221"),
        new Data(2, "group3", "222"),
        new Data(4, "group3", "24")
    };

    static List<Data> d1_for_distinct = new List<Data>()
    {
        new Data(1, "group1", "11"),
        new Data(1, "group1", "11"),
        new Data(1, "group1", "11"),
        new Data(2, "group1", "12"),
        new Data(2, "group1", "12")
    };

    static List<DataLink> lnk = new List<DataLink>()
    {
        new DataLink(1, 1),
        new DataLink(1, 2),
        new DataLink(1, 4),
        new DataLink(2, 1),
        new DataLink(2, 2),
        new DataLink(2, 4),
        new DataLink(5, 1),
        new DataLink(5, 2)
    };

static void Main(string[] args)
```

```
{
    Console.WriteLine("Простая выборка элементов");
    var q1 = from x in d1
             select x;
    foreach (var x in q1)
         Console.WriteLine(x);

    Console.WriteLine("Выборка отдельного поля (проекция)");
    var q2 = from x in d1
             select x.value;
    foreach (var x in q2)
        Console.WriteLine(x);

    Console.WriteLine("Создание нового объекта анонимного типа");
    var q3 = from x in d1
             select new { IDENTIFIER = x.id, VALUE = x.value };
    foreach (var x in q3)
         Console.WriteLine(x);

    Console.WriteLine("Условия");
    var q4 = from x in d1 where x.id > 1 && (x.grp == "group1" || x.grp == "group2")
             select x;
    foreach (var x in q4)
            Console.WriteLine(x);

    Console.WriteLine("Выборка по значению типа");
        object[] array = new object[] { 123, "строка 1", true, "строка 2" };
        var qo = from x in array.OfType<string>()
                 select x;
    foreach (var x in qo)
            Console.WriteLine(x);

    Console.WriteLine("Сортировка");
    var q5 = from x in d1 where x.id > 1 && (x.grp == "group1" || x.grp == "group2")
             orderby x.grp descending, x.id descending
             select x;
    foreach (var x in q5)
            Console.WriteLine(x);

    Console.WriteLine("Сортировка (с использованием методов)");
    var q51 = d1.Where((x) =>
        {
            return x.id > 1 && (x.grp == "group1" || x.grp == "group2");
        }).OrderByDescending(x => x.grp).ThenByDescending(x => x.id);
    foreach (var x in q51) Console.WriteLine(x);

    Console.WriteLine("Partitioning Operators");
    Console.WriteLine("Постраничная выдача данных");
    var qp = GetPage(d1, 2, 2);
    foreach (var x in qp)
            Console.WriteLine(x);

    Console.WriteLine("Использование SkipWhile и TakeWhile");
    int[] intArray = new int[] { 1, 2, 3, 4, 5, 6, 7, 8 };
    var qw = intArray.SkipWhile(x => (x < 4)).TakeWhile(x => x <= 7);
    foreach (var x in qw)
            Console.WriteLine(x);

    Console.WriteLine("Декартово произведение");
    var q6 = from x in d1
             from y in d2
```

```
                 select new { v1 = x.value, v2 = y.value };
             foreach (var x in q6)
                 Console.WriteLine(x);

         Console.WriteLine("Inner Join с использованием Where");
         var q7 = from x in d1
                  from y in d2
                  where x.id == y.id
                  select new { v1 = x.value, v2 = y.value };
         foreach (var x in q7)
                 Console.WriteLine(x);


         Console.WriteLine("Cross Join (Inner Join) с использованием Join");
         var q8 = from x in d1
                  join y in d2 on x.id equals y.id
                  select new { v1 = x.value, v2 = y.value };
         foreach (var x in q8)
                 Console.WriteLine(x);

         Console.WriteLine("Cross Join (сохранение объекта)");
         var q9 = from x in d1
                  join y in d2 on x.id equals y.id
                  select new { v1 = x.value, d2Group = y };
         foreach (var x in q9)
                 Console.WriteLine(x);


         //Выбираются все элементы из d1 и если есть связанные из d2 (outer join)
         //В temp помещается вся группа, ее элементы можно перебирать отдельно
         Console.WriteLine("Group Join");
         var q10 = from x in d1
                   join y in d2 on x.id equals y.id into temp
                   select new { v1 = x.value, d2Group = temp };
         foreach (var x in q10)
         {
                 Console.WriteLine(x.v1);
                 foreach (var y in x.d2Group)
                     Console.WriteLine("   " + y);
         }

         Console.WriteLine("Cross Join и Group Join");
         var q11 = from x in d1
                   join y in d2 on x.id equals y.id into temp
                   from t in temp
                   select new { v1 = x.value, v2 = t.value, cnt = temp.Count() };
         foreach (var x in q11)
                 Console.WriteLine(x);

         Console.WriteLine("Outer Join");
         var q12 = from x in d1
                   join y in d2 on x.id equals y.id into temp
                   from t in temp.DefaultIfEmpty()
                   select new { v1 = x.value, v2 = ((t == null) ? "null" : t.value) };
         foreach (var x in q12)
                 Console.WriteLine(x);

         Console.WriteLine("Использование Join для составных ключей");
         var q12_1 = from x in d1
                     join y in d1_for_distinct on new { x.id, x.grp } equals new { y.id,
y.grp } into details
```

```
                    from d in details
                    select d;
        foreach (var x in q12_1)
                Console.WriteLine(x);

        //Действия над множествами
        Console.WriteLine("Distinct - неповторяющиеся значения");
        var q13 = (from x in d1 select x.grp).Distinct();
        foreach (var x in q13)
                Console.WriteLine(x);

        Console.WriteLine("Distinct - повторяющиеся значения для объектов");
        var q14 = (from x in d1_for_distinct select x).Distinct();
        foreach (var x in q14)
                Console.WriteLine(x);

        Console.WriteLine("Distinct - неповторяющиеся значения для объектов");
        var q15 = (from x in d1_for_distinct select x).Distinct(new DataEqualityComparer());
        foreach (var x in q15)
                Console.WriteLine(x);

        Console.WriteLine("Union - объединение с исключением дубликатов");
        int[] i1 = new int[] { 1, 2, 3, 4 };
        int[] i1_1 = new int[] { 2, 3, 4, 1 };
        int[] i2 = new int[] { 2, 3, 4, 5 };
        foreach (var x in i1.Union(i2))
                Console.WriteLine(x);
        Console.WriteLine("Union - объединение для объектов");
        foreach (var x in d1.Union(d1_for_distinct))
                Console.WriteLine(x);

        Console.WriteLine("Union - объединение для объектов с исключением дубликатов 1");
        foreach (var x in d1.Union(d1_for_distinct, new DataEqualityComparer()))
                Console.WriteLine(x);
        Console.WriteLine("Union - объединение для объектов с исключением дубликатов 2");
        foreach (var x in d1.Union(d1_for_distinct).Union(d2).Distinct(new
DataEqualityComparer()))
                Console.WriteLine(x);

        Console.WriteLine("Concat - объединение без исключения дубликатов");
        foreach (var x in i1.Concat(i2))
             Console.WriteLine(x);
        Console.WriteLine("SequenceEqual - проверка совпадения элементов и порядка их
следования");
        Console.WriteLine(i1.SequenceEqual(i1));
        Console.WriteLine(i1.SequenceEqual(i2));


        Console.WriteLine("Intersect - пересечение множеств");
        foreach (var x in i1.Intersect(i2))
                Console.WriteLine(x);
        Console.WriteLine("Intersect - пересечение множеств для объектов");
        foreach (var x in d1.Intersect(d1_for_distinct, new DataEqualityComparer()))
                Console.WriteLine(x);

        Console.WriteLine("Except - вычитание множеств");
        foreach (var x in i1.Except(i2))
                Console.WriteLine(x);

        Console.WriteLine("Except - вычитание множеств для объектов");
        foreach (var x in d1.Except(d1_for_distinct, new DataEqualityComparer()))
```

```
                Console.WriteLine(x);

            Console.WriteLine("Функции агрегирования");
            Console.WriteLine("Count - количество элементов");
            Console.WriteLine(d1.Count());
            Console.WriteLine("Count с условием");
            Console.WriteLine(d1.Count(x => x.id > 1));

            //Могут использоваться также следующие агрегирующие функции
            //Sum - сумма элементов
            //Min - минимальный элемент
            //Max - максимальный элемент
            //Average - среднее значение
            Console.WriteLine("Aggregate - агрегирование значений");
            var qa1 = d1.Aggregate(new Data(0, "", ""), (total, next) =>
                {
                    if (next.id > 1) total.id += next.id;
                    return total;
                });
                Console.WriteLine(qa1);
            Console.WriteLine("Группировка");
            var q16 = from x in d1.Union(d2) group x by x.grp into g select new { Key = g.Key,
Values = g };
            foreach (var x in q16)
                {
                    Console.WriteLine(x.Key);
                    foreach (var y in x.Values)
                        Console.WriteLine("   " + y);
                }
            Console.WriteLine("Группировка с функциями агрегирования");
            var q17 = from x in d1.Union(d2)
                      group x by x.grp into g
                      select new { Key = g.Key, Values = g, cnt = g.Count(), cnt1 = g.Count(x =>
x.id > 1), sum = g.Sum(x => x.id), min = g.Min(x => x.id) };
            foreach (var x in q17)
            {
               Console.WriteLine(x);
               foreach (var y in x.Values)
                        Console.WriteLine("   " + y);
            }
            Console.WriteLine("Группировка - Any");
            var q18 = from x in d1.Union(d2)
                     group x by x.grp into g
                     where g.Any(x => x.id > 3)
                     select new { Key = g.Key, Values = g };
            foreach (var x in q18)
            {
                 Console.WriteLine(x.Key);
                 foreach (var y in x.Values)
                     Console.WriteLine("   " + y);
            }
            Console.WriteLine("Группировка - All");
            var q19 = from x in d1.Union(d2)
                      group x by x.grp into g
                      where g.All(x => x.id > 1)
                      select new { Key = g.Key, Values = g };
            foreach (var x in q19)
            {
                    Console.WriteLine(x.Key);
                    foreach (var y in x.Values)
                        Console.WriteLine("   " + y);
```

```
        }

        Console.WriteLine("Имитация связи много-ко-многим");
        var lnk1 = from x in d1
                   join l in lnk on x.id equals l.d1 into temp
                   from t1 in temp
                   join y in d2 on t1.d2 equals y.id into temp2
                   from t2 in temp2
                   select new { id1 = x.id, id2 = t2.id };
        foreach (var x in lnk1)
                Console.WriteLine(x);

        Console.WriteLine("Имитация связи много-ко-многим, проверка условия");
        var lnk2 = from x in d1
                   join l in lnk on x.id equals l.d1 into temp
                   from t1 in temp
                   join y in d2 on t1.d2 equals y.id into temp2
                   where temp2.Any(t => t.value == "24")
                   select x;
        foreach (var x in lnk2)
                Console.WriteLine(x);

        Console.WriteLine("Имитация связи много-ко-многим, использование let, проверка
условия");
        var lnk3 = from x in d1
            let temp1 = from l in lnk where l.d1 == x.id select l
            from t1 in temp1
            let temp2 = from y in d2
            where y.id == t1.d2 && y.value == "24"
            select y  where temp2.Count() > 0
                   //let temp2 = from y in d2 where y.id == t1.d2
                   //            select y
                   //where temp2.Any(t=>t.value == "24")
                   select x;
            foreach (var x in lnk3) Console.WriteLine(x);

        Console.WriteLine("Deferred Execution - отложенное выполнение запроса");
        var e1 = from x in d1 select x;
        Console.WriteLine(e1.GetType().Name);
        foreach (var x in e1)
                Console.WriteLine(x);
        Console.WriteLine("При изменении источника данных запрос выдает новые результаты");
        d1.Add(new Data(333, "", ""));
        foreach (var x in e1)
            Console.WriteLine(x);

        Console.WriteLine("Immediate Execution - немедленное выполнение запроса, результат
преобразуется в список ");
        var e2 = (from x in d1 select x).ToList();
        Console.WriteLine(e2.GetType().Name);
        foreach (var x in e2)
            Console.WriteLine(x);
        Console.WriteLine("Результат преобразуется в массив");
        var e3 = (from x in d1 select x).ToArray();
        Console.WriteLine(e3.GetType().Name);
        foreach (var x in e3)
            Console.WriteLine(x);
        Console.WriteLine("Результат преобразуется в Dictionary");
        var e4 = (from x in d1 select x).ToDictionary(x => x.id);
        Console.WriteLine(e4.GetType().Name);
        foreach (var x in e4)
```

```
            Console.WriteLine(x);
        Console.WriteLine("Результат преобразуется в Lookup");
        var e5 = (from x in d1_for_distinct select x).ToLookup(x => x.id);
        Console.WriteLine(e5.GetType().Name);
        foreach (var x in e5)
        {
            Console.WriteLine(x.Key);
            foreach (var y in x)
                Console.WriteLine("   " + y);
        }

        Console.WriteLine("Получение первого элемента из выборки");
        var f1 = (from x in d2 select x).First(x => x.id == 2);
        Console.WriteLine(f1);
        Console.WriteLine("Получение первого элемента или значения по умолчанию");
        var f2 = (from x in d2 select x).FirstOrDefault(x => x.id == 22);
        Console.WriteLine(f2 == null ? "null" : f2.ToString());

        Console.WriteLine("Получение элемента в заданной позиции");
        var f3 = (from x in d2 select x).ElementAt(2);
        Console.WriteLine(f3);

        Console.WriteLine("Генерация последовательностей");
        Console.WriteLine("Range");
        foreach (var x in Enumerable.Range(1, 5))
            Console.WriteLine(x);
        Console.WriteLine("Repeat");
        foreach (var x in Enumerable.Repeat<int>(10, 3))
            Console.WriteLine(x);
        Console.ReadLine();
    }

    /// <summary>
    /// Получение нужной страницы данных
    /// </summary>
    static List<Data> GetPage(List<Data> data, int pageNum, int pageSize)
    {
        //Количество пропускаемых элементов
        int skipSize = (pageNum-1)*pageSize;
        var q = data.OrderBy(x => x.id).Skip(skipSize).Take(pageSize);
        return q.ToList();
     }
  }
}
```

Постановка задачі комп‘ютерного практикуму № 4

При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:

1) Розробити структуру даних для зберігання згідно варіантів, наведених нижче. У кожному з варіантів має бути як мінімум 3-4 класи. В рамках реалізації повинні бути продемонстровані зв'язки між класами: один-до-багатьох і багато-до-багатьох.
2) Розробити як мінімум 15 різних запитів, використовуючи різні дії над множинами: групування, сортування, фільтрацію, об'єднання результатів декількох запитів в один (join, concat) та інше. Крім того, необхідно використовувати обидва можливі варіанти реалізації LINQ-запитів

(класичний варіант та з використанням методів розширення), причому запити не повинні повторюватись.

Наприклад (предметне середовище Кінофільми):

a. Вивести перелік всіх кінофільмів
b. Вивести перелік акторів, котрі грають у кінофільмах, котрі починаються з літери «А»
c. Вивести перелік всіх акторів та кінофільмів, в яких вони грають
d. Вивести перелік всіх акторів, згрупувавши дані по рокам народження
e. Вивести перелік кінофільмів, в яких хоча б у одного актора прізвище починається на літеру "А"
f. Вивести всі кінофільми, відсортувавши їх по роках
g. Вивести всіх акторів, згрупувавши по амплуа та роком народження
h. З'єднати джерела даних Кінофільм» і «Актор». Вивести назву фільму, прізвище автора, прізвище актора в головній ролі.

3) Створити програмне забезпечення, котре реалізує обробку даних з використання бібліотеки LINQ to Objects.
4) Програмне забезпечення необхідно розробити у вигляді консольного застосування на мові C#.
5) Коротко описати архітектуру проекту

Варіанти індивідуальних завдань:

1) Розробити структуру даних для зберігання інформації про книги в бібліотеці. Книга характеризується: назвою, прізвищем автора, вартістю, датою видання, видавництвом, списком інвентарних номерів (книга в кількох примірниках). У одного автора може бути декілька книг.
2) Розробити структуру даних для зберігання інформації про студентів-дипломників та їх керівників. Про студентів необхідно зберігати щонайменше наступну інформацію: ПІБ, група, дата народження, середній бал. Про керівників: ПІБ, посада. У одного керівника може бути декілька студентів-дипломників.
3) Розробити структуру даних для зберігання інформації про товари на складі. Товар характеризується: назвою, вартістю, кількістю (в штуках), списком дат надходження на склад, виробником (найменування фірми).
4) Розробити структуру даних для зберігання інформації для відстеження фінансових показників роботи пункту прокату автомобілів. В автопарк компанії входить кілька автомобілів різних марок, вартостей і типів. Кожен автомобіль має свою ціну прокату. В пункт прокату звертаються клієнти. Всі клієнти проходять обов'язкову реєстрацію - про них збирається стандартна інформація (ПІБ, адреса, телефон). Кожен клієнт може звертатися в пункт прокату кілька разів. Всі звернення клієнтів

фіксуються, при цьому по кожній угоді запам'ятовуються дата видачі та очікувана дата повернення. Перед отриманням автомобіля клієнт залишає деяку заставну суму, яка йому повністю повертається після успішного повернення автомобіля. Вартість прокату автомобіля залежить не тільки від самого автомобіля, але і від терміну його прокату, а також від року випуску.

5) Розробити структуру даних для формування навантаження викладачів кафедри. Навчальна дисципліна повинна мати наступні атрибути: назву, ПІБ викладача, форму контролю, кількість годин, код спеціальності, курс викладання. Спеціальність характеризується назвою та кодом. Передбачити, що один викладач може викладати різні дисципліни. Дисципліна з одною і тою ж назвою може бути на різних спеціальностях.
6) Розробити структуру даних для зберігання інформації про обчислювальну техніку на підприємстві. Обчислювальна техніка характеризується: назвою, вартістю, обчислювальною потужністю, списком осіб, які експлуатують обчислювальну техніку.
7) Розробити структуру даних для зберігання інформації про реєстрацію транспортних засобів. Для кожного транспортного засобу зберігається як мінімум марка авто, виробник, модель, тип кузову, рік випуску, номер шасі (VIN-код), колір, номерний знак, технічний стан, власник автомобіля, перелік водіїв, котрі мають право керувати транспортним засобом, тощо. Для власників та тих персон, котрі мають право керувати транспортним засобом, - номер прав водія, прізвище, ім'я, по батькові, дата народження, адреса реєстрації. Необхідно врахувати, що транспортний засіб може мати декілька власників (тобто бути зареєстрованим декілька разів).
8) Розробити структуру даних для зберігання інформації про квартири, котрі продаються різними агентствами нерухомості. По квартирам необхідно зберігати щонайменше наступну інформацію: адреса, поверх, площа, ціна. По агентствам – назва, адреса, ріелтери, контактні телефони кожного з ріелтерів . Передбачити, що одну й ту саму квартиру можуть продавати різні агентства нерухомості і ціна може бути різна.
9) Розробити структуру даних для зберігання інформації про проекти, що виконуються на підприємстві. По кожному проекту зберігається інформація: шифр проекту, найменування проекту, вартість робіт для виконання проекту, дата початку проекту і дата закінчення проекту, список осіб, які беруть участь в проекті.
10) Розробити структуру даних для зберігання інформації про кінотеатри міста. Для кінотеатру зберігається інформація: найменування кінотеатру, місткість (кількість місць), рік побудови, ранг кінотеатру (для перегляду відеофільмів, для перегляду широкоформатних фільмів, наявність стереоформатного обладнання, тощо).

11) Розробити структуру даних для зберігання інформації про абонементи бібліотеки. Бібліотека заробляє гроші, видаючи напрокат деякі книги, які наявні в невеликій кількості примірників. У кожної книги, яка видається в прокат, є назва, автор, жанр. В залежності від цінності книги для кожної з них визначена заставна вартість (сума, яку вносить клієнт при взятті книги напрокат) і вартість прокату (сума, яку клієнт платить при поверненні книги, отримуючи назад заставу). Вартість прокату книги залежить не тільки від самої книги, а й від терміну її прокату. У бібліотеку звертаються читачі. Всі читачі реєструються в картотеці, яка містить стандартні анкетні дані (ПІБ, адреса, телефон, категорія). Кожен читач може звертатися в бібліотеку кілька разів. Всі звернення читачів фіксуються, при цьому по кожному факту видачі книги (при наявності примірника) фіксується дата видачі та очікувана дата повернення.

12) Розробити структуру даних для зберігання інформації про тролейбусні маршрути міста. Для кожного маршруту зберігається інформація: найменування початкової і кінцевої зупинки, кількість тролейбусів на маршруті, час проїзду від початку маршруту до кінця, список номерів тролейбусів на маршруті.

13) Розробити структуру даних для зберігання інформації про бронювання номерів готелю. Номер має наступні характеристики: клас, кількість місць розміщення, додаткові опції: кондиціонер, дитяче ліжко, тощо. Клієнти бронюють номери визначеного типу на період (з дати по дату). Передбачити, що один той самий клієнт може забронювати декілька номерів на один й той самий період, або різні номери на різні періоди.

14) Розробити структуру даних для зберігання інформації про літаки. Для об'єктів зберігається наступна інформація: літак - тип літака, вантажопідйомність, максимальна дальність, розмах крил, довжина розбігу, шифр компанії; вертоліт - тип вертольоту, вантажопідйомність, максимальна висота, дальність польоту, шифр компанії; авіакомпанія - назва, місце розташування офісу, дата утворення фірми, шифр компанії.

15) Розробити структуру даних для зберігання інформації про будинки. Для об'єктів зберігається наступна інформація: житловий будинок - тип проекту, число поверхів, число під'їздів, дата побудови, шифр; район міста - назва, адреса районної адміністрації, кількість жителів, площа, шифр; список будинків - шифр району, шифр будинку.

16) Розробити структуру даних для зберігання інформації про автомобілі. Для об'єктів зберігається наступна інформація: вантажний автомобіль - марка автомобіля, вантажопідйомність, дата випуску, дата капітального ремонту, державний номер, шифр автопарку; таксі - марка автомобіля, кількість посадкових місць, дата випуску, державний номер,

шифр автопарку; автопарк - назва, адреса розміщення, площа для розміщення автомобілів, шифр

17) Розробити структуру даних для зберігання інформації про робітників підприємства та їх заробітну платню по місяцях. По кожному з робітників зберігається наступна інформація: прізвище, ім‘я, по-батькові, дата народження, табельний номер, реєстраційний номер облікової картки платника податків, освіта, спеціальність, дата початку роботи на підприємстві. В об‘єкті «Заробітна платня по місяцях» зберігається інформація про заробітну плату співробітника по місяцях з початку його роботи на підприємстві. Обидва об‘єкті зв‘язані по реєстраційному номеру облікової картки платника податків.

18) Розробити структуру даних для зберігання інформації про успішність студентів по різним дисциплінам в розрізі семестрів та курсів. Передбачити, що дисципліни можуть мати різні фіксовані форми контролю (екзамен, залік).

19) Розробити структуру даних для зберігання інформації про статті авторів, котрі опубліковані в різних журналах. Для кожного журналу зберігається наступна інформація: назва журналу, інформація про періодичність виходу журналу, дата виходу журналу, тираж журналу. Для статті зберігається інформація про її назву, авторів, журнал, в якому вона опублікована та дата надходження статті в редакцію. По кожному з авторів зберігається інформація щодо його ПІБ, організації, в якій він працює.

20) Розробити структуру даних для зберігання інформації про депозити населення в національній, так і в іноземній валюті в банку. Кожен вклад має свій строк, валюту, відсоткову ставку, крім того, початковий внесок по різним типам депозитів – різний. Крім того, банк може надавати кредити населенню в національній та іноземній валютах. Для кожного клієнта повинна зберігатися наступна інформація: ПІБ, код клієнту, реєстраційний номер облікової картки платника податків, контактні дані. Для депозитів – код депозиту, код клієнта, номер рахунка, сума вкладу, дата початку та завершення. Для кредитів – код кредиту, код клієнта, валюта, сума кредиту, дата видачі та повернення.

21) Розробити структуру даних для зберігання інформації про акторів та їх фільмографію. Крім фільмів, актори можуть грати у театрі. Кінофільми характеризуються назвою, роком випуску, жанром та режисером. Спектаклі – назвою та жанром. По кожному актору необхідно зберігати інформацію: ПІБ, амплуа, рік народження. Врахувати той факт, що в деяких фільмах актори можуть виступати і у якості режисеру.

22) Розробити структуру даних для зберігання інформації про країни. Для об'єктів зберігається наступна інформація: країни – назва, континент, загальна площа, загальна кількість населення, столиця, валюта, мова,

тощо; область – назва, площа, обласний центр; місто – назва, площа, населення; район міста - назва, адреса районної адміністрації, кількість жителів, площа.

23) Розробити структуру даних для зберігання інформації про морські перевезення. Для об'єктів зберігається наступна інформація: кораблі – назва, пасажиромісткість, пароплавство, рік побудови, тип (річкове, морське); пароплавство – назва, місце розташування (річка/море); річка – назва, довжина, максимальна ширина, кількість притоків; море – назва, площа. Врахувати наступне, що кожен корабель може експлуатуватися тільки на річці або тільки на морі. На річці і на морі можуть експлуатуватися кілька кораблів.

24) Розробити структуру даних для зберігання інформації про розклад пасажирських залізничних потягів. Для об‘єктів зберігається наступна інформація: потяг – інвентарний номер потягу, ПІБ головного по потягу, номер потягу, кількість вагонів; вагон – потяг, тип вагону (плацкарт/купейний/спальний), кількість місць; розклад – місто звідки, місто куди, дата, потяг, час прибуття, час відправлення

25) Розробити структуру даних для зберігання інформації про меню ресторану. Для об‘єктів зберігається наступна інформація: продукт – назва, кількість калорій; блюдо – назва, продукти та їх кількість; меню – дата, блюдо, вартість.

Питання до роботи:

1) Призначення бібліотеки LINQ to Objects?
2) Які структури даних можуть бути джерелами даних для бібліотеки LINQ to Objects?
3) Як реалізувати зв'язок один-до-багатьох в бібліотеці LINQ to Objects?
4) Як реалізувати зв'язок багато-до-багатьох в бібліотеці LINQ to Objects?
5) Як реалізувати перевірку умови в бібліотеці LINQ to Objects?
6) Як реалізувати сортування в бібліотеці LINQ to Objects?
7) Як реалізувати групування в бібліотеці LINQ to Objects?
8) Як в бібліотеці LINQ to Objects з використанням групування реалізувати перевірку умови, що строкові дані всіх записів (або тільки одного запису) починаються з певного символу?